工學碩士學位論文

부스팅 학습 방법을 이용한 문서 여과 시스템의 생성에 관한 연구

A Study on Learning a Text Filtering System Using Boosting Algorithm.

2000年 2月

서울大學校 大學院

컴퓨터工學科

韓 尙 潤

부스팅 학습 방법을 이용한 문서 여과 시스템의 생성에 관한 연구

A Study on Learning a Text Filtering System Using Boosting Algorithm.

指導敎授 金 榮 澤

이 論文을 工學碩士 學位論文으로 提出함

1999年 10月

서울大學校 大學院

컴퓨터工學科

韓 尙 潤

韓尙潤의 工學碩士 學位論文을 認准함

1999年 12月

委 員 長 _____ **印**

副委員長 _____ **印**

委 員 _____ **印**

초 록

문서 여과 문제(text filtering)는 어떤 문서가 특정한 주제에 속하는지의 여부를 판별하는 문제로 인터넷의 보급과 함께, 또 전자우편이나 인터넷상의 문서가 폭발적으로 증가하면서 많은 연구가 진행되고 있는 분야이다. 이 논문에서는 새로운 학습방법으로 주목받고 있는 부스팅 학습 방법을 문서 여과 문제에 적용하여 기존의 학습방법들에 비해 우수한 분류 결과를 나타내는 문서 여과 시스템을 생성하고자 한다. 이 논문에서 문서 여과 문제에 적용된 에이다부스트(AdaBoost) 학습 방법에서는 매우 단순한 분류 기준인 어떤 단어가 문서에 포함되어 있는가의 여부를 그 문서의 주제와의 관련성 여부를 판단하는 규칙으로 사용하는 가설을 여러 개 생성한 후 이들을 체계적인 방법으로 결합함으로써 보다 우수한 성능을 갖는 분류 시스템을 학습시킨다. 먼저 최종 여과 시스템을 구성하는 각 가설의 출력이 1 또는 -1이 되는 이진 가설을 사용하는 이전의 에이다부스트 알고리즘에서 출발하여 실수값을 출력하는 가설을 이용하는 에이다부스트 알고리즘을 적용함으로써 에러 감소 속도와 최종 에러율을 개선하고자 하였다. 또 기본적으로 사용한 에이다부스트 학습 방법의 성능을 향상하기 위해 각 데이터에 대한 초기 가중치를 연속 푸아송 분포(continuous Poisson distribution)에 따라 임의로 부여하여 여러 번의 부스팅을 수행한 후 그 결과를 결합하는 방법을 사용함으로써 적은 수의 학습 데이터 집합으로부터 학습을 수행할 경우 발생하는 과적화의 문제를 완화하고자 하였다. 실험 데이터로는 현재 문서 검색 분야에서 최고의 대회로 자리잡고 있는 TREC (Text Retrieval Conference)의 TREC-8 adhoc collection을 사용하였는데 이 데이터는 문서 여과 분과에서 사용하는 Financial Times 1992 1994 기사 모음 데이터이다. 실험 결과, 실수값을 갖는 가설을 사용했을 때 이진 값을 갖는 가설을 사용했을 때 보다 좋은 결과를 보였고 임의 가중치를 사용하여 여러번 부스팅을 하는 방법이 더욱 향상된 성능을 나타내었다. 다른 TREC 참가자들과의 비교도 제시한다.

주제어 : 기계학습, 문서여과시스템, 에이다부스트, 배깅, 멀티부스팅

제 목 차 례

표 차 례

그 림 차 례

제1장 서론

1.1 연구 배경

문서 분류 문제는 크게 문서 여과 문제와 문서 범주 분류 문제로 나누어 볼 수 있다. 문서 여과 문제는 어떤 문서가 특정한 사용자의 관심 분야에 관련이 있는지 그렇지 않은지를 결정하는 문제이다. 예를 들어 주식 시장의 동향에 특별한 관심을 보이는 어떤 사용자에게 배달된 전자우편이 이 사용자의 관심사에 적합한지 아닌지를 결정해 주는 것과 같은 문제이다. 따라서 이 문제는 이진 분류 문제로 볼 수 있다. 문서 여과를 위한 학습시스템에서는 단어들의 나열로 표시되는 문서와 그 문서에 대한 특정 사용자의 관심 여부 (1 또는 - 1)로 이루어진 순서쌍들의 집합을 학습 데이터로 사용하여 문서 여과를 위한 규칙을 찾아내고 이를 새로운 문서에 적용하여 그 문서가 사용자의 관심 영역에 속하는지를 판별하는 시스템을 생성한다. 한편 문서 범주 분류 문제는 이진 분류 문제가 아니라 주어진 문서가 여러 개의 분류 항목 중 어느 것에 속하는지를 결정하는 문제이다. 예를 들어 어떤 문서가 정치, 경제, 사회, 스포츠 중 어떤 주제에 속하는지 그 범주를 결정하는 것과 같은 문제이다. 따라서 이 문제는 다중 클래스 분류 문제에 속한다. 문서 범주 분류 학습시스템에서는 학습 데이터로부터 각 문서가 어떤 범주에 속하는지 판별하고 발견된 범주들간의 순위를 정해 주는 규칙을 찾아 새로운 문서에 대해 문서 범주를 결정해주는 시스템을 생성한다. 이 문제의 경우 보통 하나의 문서가 여러 개의 범주에 속할 수 있고 각 항목들간의 순위에 대한 정보가 학습 데이터에 포함되어 있지 않은 것이 일반적이기 때문에 학습이 문서 여과 문제의 경우보다 훨씬 어렵다.

이들 중 이진 분류를 수행하는 문서 여과 문제는 적절한 변환을 통해 문서 범주 분류 문제로 확장하는 것이 가능하다. 즉, 각 범주에 특정 문서가 관련 있는지의 여부를 문서 여과 시스템이 판단하고 이를 체계적으로 결합함으로써 그 문서가 어떤 범주에 속하는지와 각 범주에 대한 순위를 결정할 수 있다. 또 문서 여과 문제는 문서 범주 분류 문제의 경우보다 분류 시스템의 성능에 대한 명확한 평가 기준

을 사용할 수 있다. 문서 범주 분류 문제는 새로이 개척되고 있는 분야이기 때문에 아직 분류 시스템의 성능을 비교, 평가하기 위한 표준적 데이터 집합이나 명확한 평가 척도가 정착되어 있지 못하다. 그러나 문서 여과 문제의 경우 1992년 처음 개최되어 매년 실행되고 있는 TREC (Text REtrieval Conference)에서 데이터의 수집과 적절한 평가 척도의 개발 및 보급에 노력하여 현재 여과 시스템들에 대한 어느 정도 객관적인 비교가 가능하게 되었다.

문서 여과 문제에 대한 연구는 정보 검색 분야에서 뿐 아니라 기계학습 분야에서도 중요한 응용 문제로 많은 연구가 진행되고 있다. 이들 중에 베이지언 분류기, k 최근점 분류 방법, 인공 신경망, 규칙학습 등 현재 기계학습 분야에서 중요한 위치에 있는 학습 방법들을 적용한 연구들이 계속 발표되고 있다[lewis92, lewis94, yang94, yang95, hull96, lewis96]. 특히 최근 기계 학습 분야에서 많은 관심을 받고 있는 부스팅(boosting) 학습 방법을 문서 여과 문제에 적용한 연구 [schapire98b]에서는 매우 단순한 규칙을 결합하여 이해와 구현이 쉬우면서도 탁월한 성능을 나타내는 학습 방법을 제시하고 있다. 부스팅 학습 방법은 단순한 가설들을 결합하여 최종적 분류를 수행하는 위원회 결정 모델(decision-committee model)의 일종으로 재표본 추출에 의존하는 배깅(bagging)과 함께 최근에 주목을 받고 있는 학습 방법이다.

에이다부스트 학습 방법은 분류가 어려운 데이터들을 보다 잘 분류할 수 있도록 데이터 분포를 변경하고 변경된 데이터 분포로부터 가설을 선택하여 결합하는 방법이기 때문에 분류 에러를 구성하는 바이어스(bias), 분산(variance) 모두를 줄여주는 경향을 나타내지만[schapire98a] 특히 바이어스의 감소에 우수한 성능을 나타내는 것으로 알려져 있다[bauer99].

이러한 특성으로 인해 학습 데이터의 크기가 충분히 크지 못한 경우 과적화(overfitting) 현상이 발생하여 학습을 진행함에 따라 일반화 에러를 증가시키는 경향을 보이기도 한다. 또 에이다부스트를 문서 여과 문제에 적용한 이전의 연구 [schapire98b]에서 볼 수 있는 바와 같이 이 알고리즘은 매우 느린 학습 속도를 나타내기 때문에 이러한 문제들의 개선이 요구된다.

1.2 연구 목적

이 논문에서는 기존의 문서 여과 문제에 적용한 에이다부스트 학습 알고리즘을 개선하여 이를 표준 데이터 집합에 적용함으로써 분류 시스템의 성능을 향상하고 이를 객관적 평가 척도에 따라 다른 분류 방법들과 비교하고자 한다. 이 논문에서 사용하고 있는 에이다부스트 알고리즘이 가지는 장점은 매우 단순한 가설들을 결합하여 최종의 여과 시스템을 구성하기 때문에 사용자가 학습된 시스템에 대해 쉽게 이해할 수 있고 무엇보다도 이러한 단순한 가설을 결합하여 우수한 여과 성능을 나타내는 시스템을 학습할 수 있다는 점이다. 이 논문에서는 에이다부스트의 약점으로 지적되고 있는 학습 속도의 문제와 매우 적은 데이터에 대한 과적화 문제를 개선하고자 한다.

이 논문에서는 또 현재 문서 검색 분야에서 최고의 대회로 자리 매김하고 있는 TREC (Text REtrieval Conference)의 데이터를 사용하여 문서 여과 시스템을 학습, 평가하고 있다. TREC은 1992년에 시작되어 현재 8회에 이르고 있으며 문서 여과, 다국어 검색, 질의검색 등 7개 하위 분과로 이루어져 있다. 이 논문에서 사용하는 데이터는 제8회 대회에서 문서 여과 분과의 참가자들이 사용한 데이터로, 각 문서에 대한 사용자 50명의 관심 여부를 표시한 데이터이다. 이 데이터는 한 사용자의 관심 여부가 표시된 문서의 갯수가 매우 적기 때문에 여과 시스템의 학습이 어렵다는 특성을 갖는다. 이 데이터에 에이다부스트 학습 방법을 적용할 경우 적은 학습 데이터로 인해 위에서 언급한 과적화 문제가 발생하기 때문에 이번 논문에서는 이를 개선하기 위한 방법을 모색하고자 한다.

1.3 논문의 구성

이 논문에서는 먼저 2장에서 위원회 결정 알고리즘들의 이론적 기초를 제공해 주는 PAC(Probably Approximately correct) 학습 모델과 약 학습 모델에 대해 살펴보고, 부스팅 학습방법과 또 하나의 위원회 모델인 배깅 학습 방법에 관한 이전 연구를 살펴보기로 한다. 3장에서는 문서 여과 문제에 대한 정의를 제시하고 문서 여과 시스템 성능 평가 방법에 대해 살펴보기로 한다. 4장에서는 이 논문에서 사용하는 학습 방법을 자세히 설명하고, 5장에서는 실험에 사용한 데이터에 대한 설명과 실험 방법 및 실험 결과를 상술하며, 마지막 장에서는 지금까지의 결과를 요약하고 새로운 연구 주제들을 제시하고자 한다.

제2장 부스팅 및 배깅 학습 방법

2.1 강 학습 모델과 약 학습 모델

컴퓨터 학습 이론 분야에서는 학습 방법과 에러와의 관계에 기초한 두가지 학습 모델이 제안되었는데, 강 학습 모델(strong learning model)과 약 학습 모델(weak learning model)이 그것들이다. Valiant는 1984년 이진 개념의 학습 가능성에 대한 연구를 통해 PAC (Probably Approximately Correct)이라는 이론적 학습 모델을 제시하였다[neuralnet]. 어떤 데이터 x가 $x = (a_1, \cdots, a_n)$ 와 같이 속성들의 벡터로 표시되고 이러한 모든 가능한 데이터들의 집합을 X 라고 하자. 또 가능한 모든 개념들의 집합을 $C, c(x) \in \{0, 1\}$를 데이터 x가 나타내는 개념이라고 하자. 이때 학습 시스템은 확률분포 D에 따라 X 로부터 생성된 학습 데이터를 사용하여 모든 데이터 x에 대해 $c(x) = h(x)$가 되는 $h(x)$를 찾아내는 것을 최종의 목표로 한다. 다시 말해 전체 데이터 공간에 대해 에러가 0이 되는 가설 h를 찾는 것이 학습시스템의 목표가 되는 것이다. 그러나 이러한 목표는 지나치게 이상적인 것으로, 데이터가 생성되는 분포를 미리 알 수 없고, 학습시 사용 가능한 데이터는 데이터 공간내의 모든 데이터가 아니라 그 중의 일부에 불과하기 때문에 가능한 모든 데이터에 대한 에러가 0인 가설을 학습한다는 것은 불가능하다. 그 대신, 학습된 가설의 에러 범위 상한값이 작은 상수 ε이 되고, 임의로 데이터를 추출하여 가설에 적용했을 때 에러가 발생할 확률이 최대 작은 상수 δ가 되도록 가설을 실행 가능한 시간안에 학습하는 것을 목표로 하는 모델이 PAC 학습 모델이다. PAC 모델에서는 모든 입력 데이터 분포에 대해 아주 작은 상수 에러 상한값을 요구하기 때문에 이를 강 학습 모델이라고도 한다.

이에 비해 약 학습 모델은 에러 상한에 대한 조건을 완화하여 임의로 추측한 경우보다 약간 좋은 정확도를 갖는 가설의 학습만을 요구한다[neuralnet]. 즉 이진

분류 문제의 경우 에러가 1/2보다 약간 작도록 가설을 학습하는 것만을 요구하고 있다. 약 학습 모델과 관련하여 중요한 문제는 "과연 강 학습 모델과 약 학습 모델이 같은 학습 능력을 갖는가"하는 질문에 대한 답을 찾는 것이었는데 이에 대해 Schapire는 1990년 그의 논문 [schapire90]에서 새로운 학습 알고리즘을 제안하여 약 학습 모델이 강 학습 모델과 동등한 학습 능력을 가짐을 입증하였다. 이 학습 방법은 최초의 부스팅 학습 방법으로 다음과 같이 학습을 수행한다. 먼저 n개의 데이터를 사용하여 하나의 가설을 학습한다. 다음으로 새로운 데이터를 학습된 가설에 적용하여 옳게 예측된 데이터와 틀리게 예측된 데이터의 비율이 거의 1이 되도록 n개의 데이터를 추출한다. 추출된 데이터를 사용하여 다시 하나의 가설을 생성하고 첫번째 학습된 가설과 두번째 학습된 가설에 또 다른 데이터를 적용하여 두 가설의 예측이 일치하지 않는 데이터 n개를 다시 추출한다. 이 데이터를 사용하여 세번째 가설을 학습하고 이 세 가설의 투표를 통해 새로운 데이터가 속하는 개념을 결정한다. 이 방법에서는 두번째, 세번째 가설을 학습해 감에 따라 틀리기 쉬운 데이터들을 추출함으로써 보다 분류하기 어려운 데이터에 집중하여 학습하는 경향을 나타낸다. 이 방법을 통해 학습된 세 가설을 사용하여 개념을 예측하는 경우 각 가설의 에러 상한을 ε이라 할때 최종 시스템의 에러는 $3\varepsilon^{2}-2\varepsilon^{3}$을 상한값으로 갖으며 [schapire90] 이 방법을 재귀적으로 여러 번 사용하면 에러를 임의의 작은 값으로 줄일 수 있다. 따라서 약 학습 모델도 강 학습 모델과 같은 학습 능력을 갖는 것을 보일 수 있다.

2.2 에이다부스트 학습 방법

위에서 언급한 데이터를 여과하는 방식의 부스팅 학습 방법은 그 뒤의 부스팅 학습 방법에 비해 메모리를 적게 사용한다는 장점이 있기는 하지만 세 가설을 학습하기 위해서는 상당히 많은 데이터가 필요하기 때문에 실제 문제에 적용하기 어렵다. 이러한 문제를 개선하기 위해 제안된 새로운 부스팅 학습 방법이 에이다부스트 학습 알고리즘이다[freund96a]. 이 알고리즘 역시 약 학습 모델에 기반하고 있으며 여러개의 가설들로 이루어진 위원회를 구성하여 분류를 수행한다. 그러나 에이다부스트는 이전의 부스팅 방법과는 달리 학습된 가설의 에러율에 따라 원래의 데이터로부터 재표본 추출을 수행하여 새로운 분포를 갖는 데이터 집합을 생성한 후 이를 사용하여 다시 새로운 가설을 학습하고 위원회를 구성한다. 표 1에서는 첫번째 에이다부스트 학습 알고리즘인 AdaBoost.M1 알고리즘을 보여주고 있다[freund96a].

이 알고리즘에서는 초기 데이터의 분포가 균일(uniform)하다고 가정하고 각 데이터에 대한 가중치를 동일하게 $1/m$로 초기화한다. 이러한 분포를 갖는 데이터 집합으로부터 가설을 학습하여 그 가설의 에러를 계산하고 에러가 $1/2$이상이면 원래의 데이터 집합으로부터 부트스트랩하여 새로운 분포의 데이터 집합을 생성한다. 에러가 $1/2$보다 작으면 데이터 분포와 관련이 있는 각 데이터의 가중치를 조절하고 정규화하여 새로운 분포를 시뮬레이트한다. 가중치의 갱신은 가설이 옳게 예측한 데이터에 대한 가중치는 감소시킴으로써 다음번의 가설 학습에서 보다 큰 가중치를 갖게 된 데이터, 즉 가설이 틀리게 예측한 데이터에 집중하도록 한다. 최종적으로 모든 학습된 가설들간의 투표는 가중치 $\log(1/\beta)$를 곱하여 합해지는데 이것은 적은 에러를 갖는 가설의 예측이 전체 시스템의 결정에 보다 중요한 영향을 미치게 하는 역할을 한다.

입력 : m개의 데이터를 갖는 학습 데이터 집합 S,
가설을 학습하는 알고리즘 I,
학습 수행 횟수 T

1 $s' = S$ (각 데이터에 대한 가중치를 $\frac{1}{m}$으로 설정)

2 For $i = 1$ to T {

3 $C_i = I(S')$

4 $\varepsilon_i = \frac{1}{m} \Sigma_{x_j \in S' : C_i(x_j) \neq y_j} weight(x)$ (학습 데이터 집합에 대한 가중 에러)

5 $\varepsilon_i > \frac{1}{2}$이면 S에서 부트스트랩을 하여 S'를 생성하고 각 데이터의 가중치를 $\frac{1}{m}$로 설정한 후 단계 3으로 돌아간다.

6 $\beta_i = \frac{\varepsilon_i}{1 - \varepsilon_i}$

7 $x_j \in S'$인 각 데이터에 대해, $C_i(x_j) = y_j$ 이면 $weight(x_j) \cdot \beta_i$로 가중치 수정

8 S'에 대한 가중치의 합이 1이 되도록 가중치를 정규화한다.

9 }

10 $C^*(x) = \arg\max_{y \in Y} \Sigma_{i:C_i(x)=y} \log(\frac{1}{\beta_i})$ 출력 : C^*

표 1 AdaBoost.M1 학습 알고리즘

이 학습 모델의 최종 에러는 n번째 가설 학습시의 데이터 분포 (가중치)를 D_n, n번째 가설의 에러를 ε_n ($\varepsilon_n < (1/2)$), $\gamma_n = (1/2) - \varepsilon_n$라 하면 다음과 같은 상한값을 갖는다.

$$\frac{|\{i:C^*(x_i)\neq y_i\}|}{m} \leq \Pi_{n=1}^{T}\sqrt{1-4\gamma_n^2} \leq \exp(-2\Sigma_{n=1}^{T}\gamma_n^2)$$

이 상한값은 새로운 가설을 추가함에 따라 학습 에러가 지수적으로 감소함을 보이고 있다. 실제에 있어 에이다부스트는 이러한 이론적 에러 상한값 보다 적은 학습 에러를 나타내며 일반화 에러 역시 학습 에러와 함께 감소하는 경향을 보인다. 이러한 경향에 대한 설명으로 Schapire는 에이다부스트 알고리즘이 최종 분류기의 경계폭(margin)을 최대화한다는 점을 들고 있다[schapire98a]. 여기서 경계폭이란 옳은 클래스에 대한 가설들의 투표의 가중합과 틀린 클래스에 대한 가설들의 투표 가중합 사이의 차를 말하는 것으로 새로운 가설을 학습하여 위원회에 추가함에 따라 이 경계폭이 증가하게 되므로써 학습에러와 일반화 에러 모두 감소하는 현상을 보이게 된다는 것이다. 이에 대해 Breiman은 에이다부스트에 비해 경계폭은 더 증가시키지만 에러 감소면에서는 좋은 성능을 나타내지 못하는 예를 들어 경계폭을 증가시키는 것 이외에 에이다부스트의 에러 감소에 영향을 주는 요소가 있음을 암시하고 있다 [breiman97].

에이다부스트 알고리즘이 처음 소개된 이후 이를 개선하고 다양한 응용 분야에 적용하려는 많은 시도가 이루어져 왔다. 먼저 이 알고리즘에서 이용하는 가설을 학습하는 방법으로 C4.5 [quinlan96], perceptron [freund99], Naive Bayes 분류기 [bauer99] 등의 다양한 방법들이 적용되었으며, 게임 이론 [freund96b], 자연언어처리 [abney99], 문서 여과 및 범주 분류 [schapire98b, schapire99b] 등의 여러 응용 분야에 적용되었다. 또 학습된 가설의 투표시 가중치인 α 값을 결정하는 방법 [freund97], 학습된 가설이 -1 또는 1과 같은 이산값을 출력하는 경우뿐 아니라 실수값을 출력할 수 있도록 확장하는 방법 등도 연구되었다[schapire99a]. 이중에서 문서 여과 문제에 적용된 알고리즘과 실수값을 출력하는 가설을 사용하는 에이다부스트 알고리즘은 4장에서 자세히 다루기로 한다.

이밖에도 에이다부스트의 성질에 관련된 많은 실험적 연구가 발표되었는데, 이들 중 몇가지를 살펴보면 다음과 같다. 분류 시스템의 에러는 바이어스와 분산의

합으로 나누어 생각할 수 있는데, 에이다부스트는 이론상 주로 바이어스를 줄이는 메카니즘을 사용하지만 [friedmam] 실제에 있어서는 바이어스와 분산을 모두 줄여 주는 것으로 알려져 있다[breiman96b, schapire98a]. 또 하나의 가설을 위원회에 추가함에 따라 나타나는 에러의 감소율은 학습이 계속되면서 점차 줄어든다고 한다[freund96]. 즉, 학습의 초기에는 하나의 가설을 위원회에 추가함에 따라 커다란 에러 감소 효과를 보이나 학습이 거듭됨에 따라 에러 감소의 폭은 계속 줄어들어 가는 현상을 나타낸다.

2.3 배깅 학습 방법 및 그 변종

약 학습 모델에 기반한 다른 학습 방법 중 많은 연구가 수행되고 있는 배깅(bagging) 학습 방법이란 것이 있는데, 이 학습 방법도 부스팅과 같이 서로 다른 데이터 분포로부터 가설들을 학습하여 이들로 구성된 위원회에서의 투표에 의해 최종 분류를 수행한다. 배깅이 부스팅과 구별되는 것은 학습된 가설의 에러에 따라 새로운 분포를 갖는 데이터를 생성하여 새로운 가설을 학습하는 것이 아니라 각 가설을 학습할 때 임의로 학습 데이터 집합에 대한 재표본 추출을 수행하여 생성된 데이터 집합으로부터 새로운 가설을 학습한다는 점에서이다. 즉 배깅에서는 n개의 데이터를 갖는 집합으로 부터 임의 복원 추출을 통해 새로운 분포의 데이터 집합을 만들고 이 데이터 집합을 사용하여 새로운 가설을 학습한 후 이와 같은 과정을 반복함으로써 여러개의 가설을 학습하고 이들 가설들로 예측을 위한 위원회를 구성한다[breiman96a]. 배깅은 재표본 추출을 통해 데이터의 분포만을 변경하여 반복적으로 가설을 학습하는 것이기 때문에 가설을 학습하는 시스템은 데이터의 분포에 민감한 불안정한(instable) 방법이 좋다[breiman96a]. 또 배깅은 에이다부스트가 바이어스와 분산 모두를 줄여 에러를 감소시키는데 반해 주로 분산을 줄이는 역할을 한다[breiman96b]. 특히 분산을 감소시키는 능력에 있어서는 에이다부스트를 능가하고 있다[bauer99].

또 에이다부스트와 배깅의 장점을 결합하여 분류 성능을 향상하려는 연구도 발표되었는데, 멀티부스팅 (MultiBoosting)이라고 하는 이 방법에서는 에이다부스트 방법을 사용하여 소위원회를 구성하고 각 소위원회마다 표본 재추출을 하여 새로운 데이터 분포에 대한 학습을 수행한 후 학습된 가설들간의 투표를 통해 분류를 수행한다[webb]. 이 방법을 사용하여 실험한 결과 바이어스는 거의 에이다부스트 만큼 감소시키고 분산은 배깅 만큼의 감소율을 얻는 것을 볼 수 있다.

제3장 문서 여과 문제

어떤 사용자에게 매일 수천통의 전자우편이 배달된다고 하자. 이 사용자는 이렇게 많은 메일을 모두 확인할 수도 없고 확인하고 싶지도 않을 것이다. 이 사용자가 관심을 가지는 전자우편에 대한 경향을 찾아 새로 발송되어 오는 전자우편 중 사용자가 관심을 갖는 것만을 걸러 주는 시스템이 있다면 많은 시간과 노력을 절약할 수 있을 것이다. 문서 여과 시스템은 주어진 문서의 집합과 각 문서에 대한 주제와의 관련성 여부를 표시하는 값(1 또는 -1)으로부터 규칙을 발견하여 새로운 문서에 대해 주제와의 관련성 여부를 판단해 주는 시스템이다. 문서 여과 시스템의 입력으로는 문서를 기술하는 속성들의 벡터인 d와 그 문서가 사용자의 관심 분야에 속하는지 관련성 여부를 나타내는 y의 순서쌍집합이 제공되고 시스템은 문서와 관련성 여부 사이의 규칙을 자동적으로 학습하여 새로운 문서에 대한 관련성 여부를 결정하는 규칙을 생성하는 시스템이다. 즉 $<(d_1, y_1), \ldots, (d_m, y_m)>$ $(d_i \in X, y \in \{-1, 1\})$로 표시되는 학습 데이터에 대해 학습을 수행하여 $C: X \rightarrow \{-1, 1\}$ 인 함수 C를 출력하는 것이다. 이 시스템의 목표는 생성된 규칙의 분류 에러가 최소가 되도록 하는 것이다. 분류 에러는 $\varepsilon = 1/m \sum_{i:C(d_i) \neq y_i} 1$으로 정의되고 이때 여과 시스템에서는 이 에러를 최소화하도록 규칙을 학습하게 된다.

현재 문서 검색 분야에서 최고의 대회로 자리하게 된 TREC (Text REtrieval Conference)의 7개 과제 중 문서 여과 과제에서는 이 문제를 적응적 여과 문제, 배치 여과 문제, 라우팅 여과 문제와 같은 세개의 하위 과제로 구분하고 있다. 적응적 여과 과제에서는 여러 개의 주제와 각 주제에 관련이 있는지 없는지의 여부가 표시된 문서들의 집합으로부터 먼저 각 주제와 관련된 프로필 (분류 규칙)을 학습한다. 그 후에 새로운 문서가 들어오면 학습된 프로필을 사용하여 관련성 여부를 판단하고 이 문서에 대한 관련성 여부가 표시되어 있으면 이를 프로필을 향상시키는데 사용한다. 이것은 실제 상황속에서 사용자의 피드백을 모방하는 역할을 한다. 최종적으로 학습된 프로필을 사용하여 각 주제와 관련있는 문서를 지정된 수만큼

찾아주면 되는 과제이다. 한편 배치 여과 과제에서는 주어진 모든 학습 데이터를 사용하여 분류 규칙을 학습하고 이를 테스트 데이터에 적용하여 그 수행 능력을 평가한다. 마지막으로 라우팅 과제에서는 학습 데이터를 사용하여 각 문서에 대해 주제와의 관련성을 나타내는 점수를 출력하는 규칙을 학습한 후 새로운 문서에 대해 이 점수를 계산한 후 가장 높은 점수를 얻은 1000개의 문서를 제시한다. 이 논문에서는 전통적 문서 여과 문제에 대한 대회인 배치 여과 과제의 경기 규칙과 데이터를 사용하여 실험하고 참가자들의 성능과 비교한다.

문서 여과 시스템의 성능을 평가하는 기준으로 여러가지 방법이 사용되어 왔다. 그 중 분류 에러는 가장 단순한 평가 기준으로 최종 분류 규칙들간의 위에서 제시한 ε값을 비교하는 것이다. 그러나 이 기준은 대체로 어떤 주제와의 관련성이 있는 문서가 관련성 없는 문서보다 훨씬 적은 문서 여과 문제의 특성상 모든 문서를 관련없는 것으로 판단하는 시스템의 에러가 매우 작게 나타나는 것과 같은 문제를 갖는다. 다음으로 문서 검색 분야에서 많이 사용되는 회수율(recall)과 정확률(precision)과 관련된 척도가 있다. 다음과 같은 행렬을 생각해 보자.

	관련있는 문서 수	관련없는 문서 수
검색된 문서	R_+	N_+
검색안된 문서	R_-	N_-

여기서 R_+는 관련있는 것으로 표시되고 학습된 규칙에 의해 관련있는 것으로 예측된 문서의 수, R_-는 관련있는 것으로 표시되고 관련없는 것으로 예측된 문서의 수, N_+는 관련없는 것으로 표시되고 관련있는 것으로 예측된 문서의 수, N_-는 관련없는 것으로 표시되고 관련없는 것으로 예측된 문서의 수이다. 회수율 r은 $r = \frac{R_+}{R_+ + R_-}$이고 정확률 p는 $p = \frac{R_+}{R_+ + N_+}$으로 정의된다.

즉 회수율은 관련있는 문서중 여과 시스템이 관련있는 것으로 분류한 문서의 비율이고 정확률은 여과 시스템이 관련있는 것으로 예측한 문서 중 실제 관련있는

문서의 비율을 나타낸다. 이 회수율, 정확률과 관련된 척도 중에는 동등점(break-even point)이 있는데 이것은 회수율과 정확률이 같아지는 점을 찾아 이 값을 평가 척도로 사용하는 것인데 높은 동등점을 갖는 검색 시스템이 보다 좋은 성능을 나타내는 것으로 평가된다. 그러나 이 동등점 척도는 이를 계산하기 위해 사용하는 인터폴레이션이 시스템이 출력할 수 없는 값을 산출할 수 있고, 이 동등점이 사용자의 요구에 적합한 검색 시스템을 찾는데 별 의미를 갖지 못하는 등 여러 문제점을 갖는다[schapire98b]. 또 검색 시스템의 평가에 많이 사용되는 척도가 F_1 척도인데 이것은 $F_1 = \frac{2rp}{r+p}$ 로 정의된다. 이 척도 역시 사용자의 해석이 어렵다는 단점을 가지지만 동등점 척도보다는 바람직한 평가 기준으로 사용될 수 있다.

이들 척도의 공통적 문제점은 이 값을 알아내기 위해서는 회수율을 먼저 계산해야 하는데 회수율은 모든 데이터를 본 후에야 결정할 수 있다는 것이다. 따라서 온라인 시스템에 대한 평가가 어렵고 사용자가 일정 기간 동안의 검색 성능을 평가할 수 없다는 문제를 갖는 것이다. 이러한 문제를 해결하기 위해 TREC에서는 유틸리티 척도를 사용한다. 유틸리티 척도는 $aR_+ + bN_+ + cR_- + dN_-$ 로 정의하고 이 때 a, b, c, d는 R_+, N_+, R_-, N_- 의 상대적 중요성을 나타내는 값이다. 유틸리티 값이 클수록 여과 시스템의 성능은 좋은 것으로 평가된다. 현재 TREC에서 사용되는 유틸리티 척도로는 $LF1 = 3R_+ - 2N_+$ 와 $LF2 = 3R_+ - N_+$ 가 있다. 이 두 유틸리티 척도 모두 여과 시스템이 관련성을 갖는다고 예측한 문서가 실제로 관련성을 갖는 경우에 대해 보다 많은 점수를 주고 이 예측이 틀린 경우에는 이보다 작은 벌점을 주고 있다. 여과 시스템은 이 유틸리티 값이 최대가 되게 하는 규칙을 생성함으로써 분류 에러를 최소화할 수 있으며 모든 문서에 대해관련이 없다고 예측함으로써 작은 에러를 나타내는 시스템보다 관련성 있는 문서를 정확히 예측하는 규칙을 선호하게 한다. 이 논문에서는 이들 척도 중 에러, 회수율, 정확률, $LF1$ 등의 척도를 실험 결과의 비교를 위해 사용하고 있다.

제4장 연구방법

이 논문에서는 우수한 문서 여과 시스템을 학습하기 위해 에이다부스트 학습 방법을 사용하고 있다. 이 논문에서 사용하는 에이다부스트는 Schapire가 문서 여과 문제에 적용한 버전에 기초하고 있다[schapire98b]. 먼저 이 알고리즘을 기술하고 이의 개선사항들을 살펴보기로 한다.

4.1 문서 여과 문제에 적용된 에이다부스트

이 알고리즘에서는 어떤 단어 (또는 어구)가 문서에 포함되어 있는지의 여부를 가설 (규칙)로 사용한다. 이처럼 단순한 규칙으로 분류를 수행하면 물론 많은 문서를 잘못 분류하게 되지만 이러한 가설들을 여러개 선택하여 체계적인 방법으로 결합함으로써 하나의 높은 정확도를 갖는 가설을 생성할 수 있게 된다. 이 때 학습된 단순한 가설을 약 가설이라 하고 여러개의 약 가설을 결합한 것을 최종 가설이라고 한다. 표 2에 이 알고리즘이 제시되어 있다.

표 2의 알고리즘에서 $D_s(i)$는 s번째 라운드 (s번째 약 가설의 학습시)에서 i번째 문서가 가지는 가중치를 나타내는 것으로 D는 학습 데이터에 대한 분포를 의미한다. 따라서 D는 $D_{s(i)}$가 양수이고 $\sum_{i=1}^{N} D_{s(i)} = 1$라는 제약조건을 갖는다. 또 학습 데이터의 초기 분포 D_1은 $D_{1(i)} = 1/N$의 균일 분포로 초기화된다. 이 데이터 분포 D_s의 갱신은 약 가설 학습 알고리즘에 의해 학습된 약 가설 h_s의 에러 ε_s를 계산한 후 h_s의 문서 d_i에 대한 예측이 옳으면 $e^{-\alpha_s}$를 곱하고 틀린 경우에는 e^{α_s}를 곱함으로써 $D_{s+1}(i)$의 값을 얻는다. 여기서 α_s는 $1/2\ln\left(\frac{1-\varepsilon_s}{\varepsilon_s}\right)$으로 정의되어 있다. (이 값의 결정 과정은 [freund97]를 참고) 이 때 $\varepsilon_s < 1/2$이면 $\alpha_s > 0$가 되고 $e^{\alpha_s} > 1$, $e^{-\alpha_s} < 1$이므로 h_s가 문서 d_i를 잘못 분류한 경우에는

$D_{s+1}(i)$의 값이 증가하고 옳게 분류한 경우에는 이 값이 감소하게 된다.

입력 : N개의 문서와 특정 주제와의 관련성 여부:

$<(d_1, y_1), \ldots, (d_N, y_N)>$ $(y_i \in \{-1, 1\})$

반복 회수 T (학습될 약 가설의 갯수와 일치)

초기화 : 데이터의 분포를 나타내는 가중치 D를 $D_1(i) = \frac{1}{N}$로 초기화

$s = 1$ to T 까지 반복

1 학습 데이터와 데이터 분포 D_s를 인자로 약 가설 학습 알고리즘을 호출하여 하나의 약 가설 h_s를 얻는다.

2 h_s의 에러 $\varepsilon_s = \sum_{i:h_s(d_i) \neq y_i} D_s(i)$를 계산한다.

3 $\alpha_s = \frac{1}{2} \ln(\frac{1-\varepsilon}{\varepsilon})$

4 데이터 분포를 다음과 같이 갱신한다.

$$D_{s+1}(i) = \frac{D_s(i)\exp(-\alpha_s y_i h_s(d_i))}{Z_s} = \frac{D_s(i)}{Z_s} \cdot \begin{cases} e^{-\alpha_s} & \text{if } h_s(d_i) = y_i \\ e^{\alpha_s} & \text{if } h_s(d_i) \neq y_i \end{cases}$$

여기서 Z_s는 데이터 분포의 정규화를 위한 값이다.

다음과 같은 최종 가설을 출력한다.

$h_{fin}(d) = sign(\sum_{i=1}^{T} \alpha_s h_s(d))$

표 2 문서 여과를 위한 에이다부스트 알고리즘

각 라운드마다 학습된 약 가설은 단어 (또는 어구)가 문서에 나타나는가의 여부에 따라 주제와의 관련성 여부를 나타내는 1 또는 -1 값을 출력한다. 즉 약 가설 학습 알고리즘에서 하는 일은 가능한 모든 단어 중 분류 에러를 가장 작게 하는 하나의 단어를 선택하는 것이다. 보다 정확히 말하면 약 가설 시스템에서는 먼저 문서에 나타나는, 의미있는 모든 단어에 대해 $\varepsilon = \sum_{i:h_s(d_i) \neq y_i} D_s(i)$를 계산하고 이들

중 ε이 가장 작은 단어를 분류 규칙 h_s로 선택한다. 이 약 가설은 선택된 단어가 문서에 포함되어 있으면 1을, 포함되어 있지 않으면 -1을 출력한다. 그런데 어떤 단어의 에러가 매우 크다면 (0.9) 위와 같이 가설을 학습할 경우 이 단어는 결코 약 가설로 선택되지 않을 것이다. 그러나 실제로 이 단어는 문서가 그 주제와 관련성이 없음을 나타내주는 중요한 정보가 된다. 따라서 실제 가설의 결정에 있어서는 $|\varepsilon - (1 - \varepsilon)|$가 최대가 되게 하는 단어를 선택한다. 이와 같이 학습된 약 가설들은 다음과 같은 방법을 통해 최종 가설로 결합된다.

$$h_{fin}(d) = sign(\sum_{i=1}^{T} \alpha_s h_s(d))$$

이것은 학습된 가설들간의 가중치가 부여된 투표로 볼 수 있다. 여기서 α_s는 각 약 가설 h_s가 최종 투표시 가지는 영향력을 나타낸다. 즉 1/2에 가까운 에러를 갖는 가설 h_s는 가중치 α_s가 0에 가까운 값을 갖고 이 값은 h_s의 분류가 전체 투표에 적은 영향만을 미치게 한다. 에러가 0에 가까운 작은 값이면 α_s는 큰 양수값이 되고 가설의 예측은 최종 투표에 많은 영향을 주게 되며, 에러가 1에 가까운 값을 갖게 되면 α_s가 절대값이 매우 큰 음수값이 되어 최종 결과에 부정적 영향을 크게 미치게 된다.

이 알고리즘을 제안한 논문 [schapire98b]에서는 이 알고리즘의 성능에 대한 실험 결과를 현재 최고의 성능을 보이는 것으로 알려진 Rocchio 방법 및 그 밖의 여러 알고리즘과 비교하여 제시하고 있다. 여기서 에이다부스트는 다른 방법들보다 우수한 성능을 보이는 것으로 나타나고 있는데 특히 학습 데이터가 충분히 많은 문제에서 우수한 성능을 보인다. 그러나 이 방법은 분류 에러를 충분히 감소시키는데 걸리는 시간이 Rocchio 방법에 비해 매우 길다는 문제점을 갖는다. 따라서 이 논문에서는 다음에서 논의할 방법을 사용하여 에러 감소 속도를 향상하고자 한다.

4.2 실수 가설을 이용한 에이다부스트

Schapire와 Singer는 그들의 논문 [schapire99a]에서 학습된 약 가설이 실수값을 가질 수 있도록 에이다부스트 알고리즘을 확장하고 있다. 즉 가설 $h:X \rightarrow R$는 1 또는 -1인지에 대한 판단만 하는 것이 아니라 이 판단에 대한 확신 정도를 나타내는 실수값을 출력하는 가설을 사용할 수 있게 한 것이다. 여기서 h의 부호가 양수이면 1, 음수이면 -1을 나타내는 것이고 $|h|$는 이 예측에 대한 확신 정도를 나타낸다. 이러한 가설을 이용하는 경우 이진 가설을 사용할 때와는 다른 에러 계산 방법이 필요하게 된다. 왜냐하면 1 또는 -1을 출력하는 가설의 경우 단어의 포함 여부와 문서의 주제와의 관련성 여부만을 고려하여 에러를 계산하면 되지만 실수값을 출력하는 가설을 사용할 경우 단어의 포함 여부 뿐 아니라 이를 통한 판단의 확신 정도에 대한 고려도 에러의 계산에 포함되어야 하기 때문에 기존의 에러 계산식을 그대로 사용할 수 없다. 우선 이러한 가설을 이용하는 에이다부스트 알고리즘을 다음 표에 제시하였다. 여기서 $f(x) = \Sigma_{s=1}^{T} \alpha_s h_s(x)$, 어떤 명제 π에 대해 $I(\pi)$를 π가 참이면 1, 거짓이면 -1을 갖는 함수로 정의하면 최종 가설 $H(x) = sign(F(x))$의 에러의 상한은 다음과 정리와 같이 표시될 수 있다.

정리 : 최종 가설 H의 에러는 다음과 같은 상한값을 갖는다.

$\frac{1}{m}|(i:H(x_i) \neq y_i)| \leq \Pi_{s=1}^{T} Z_s$(증명은 [schapire99a]를 참조하라.) 이와 같은 에러의 상한값은 α_s의 값과 약 가설을 선택하는데 있어 중요한 의미를 갖는다. 즉, 직접 에러를 최소화하는 약 가설이나 α_s 값을 선택하는 대신에 Z_s를 최소화하는 가설이나 α_s를 선택함으로써 에러를 줄여주는 효과를 얻을 수 있는 것이다. 이는 이산 분류 문제에서 실수값을 갖는 가설을 사용할 경우 에러의 측정에서 발생하는 문제를 고려할 필요없이 h_s나 α_s값을 선택할 수 있게 해 준다.

입력 : 학습 데이터 $<(x_1, y_1), \cdots, (x_m, y_m)>$ $(x_i \in X, y_i \in \{-1, 1\})$

반복 횟수 T

초기화 : 데이터에 대한 가중치 $D_{1(i)} = 1/m$.

For s = 1 to T

1 D_s를 사용하여 약 가설 $h_s : X \rightarrow R$를 학습

2 $\alpha_s \in R$를 선택

3 다음과 같이 데이터 분포 갱신:

$D_{s+1}(i) = \frac{D_s(i)}{Z_s} \exp(-\alpha_s y_i h_s(x_i))$ 여기서 Z_s는 데이터 분포를 정규화하기 위한 값으로 다음과 같이 정의된다.

$Z_s = \sum_i D_s(x_i) \exp(-\alpha_s h_s(x_i) y_i)$

다음과 같은 최종 가설을 출력한다.

$H(x) = sign(\sum_{s=1}^{T} \alpha_s h_s(x))$

표 3 실수값을 출력하는 가설을 이용한 에이다부스트

이 논문에서는 α_s의 값을 1로 고정하고 약 가설만을 이 에러 상한을 사용하여 선택하였다. 즉, $Z_s = \sum_i D_{s(i)} \exp(-y_i h_s(x_i))$를 최소화하는 h_s를 찾으면 되는 것이다. 이를 위해 먼저 어떤 단어가 포함된 문서의 집합을 x_1, 포함되어 있지 않은 문서들의 집합을 X_0으로 표시하고, 즉 $x \in X_j, j \in \{0, 1\}$,에 대해 $c_j = h(x)$로 정의한다. 또 어떤 단어가 문서에 포함되어 있는가의 여부, 그 문서가 주제와 관련이 있는가의 여부에 따라 결정되는 문서 집합 크기의 전체 문서에 대한 비율을 나타내는 $w_b^j = \sum_{i:x_i \in X_j \wedge y_i = b} D(i)$를 정의한다. 여기서 b는 문서가 주제와 관련성을 갖

는지 여부를 나타내는 값으로 + 또는 -로 표시한다. 이를 이용하여 Z를 다시 쓰면

$$Z = \sum_j \sum_{i:x_i \in X_j} D(i) \exp(- y_i c_j) = \sum_j (w_+^j e^{- c_j} + w_-^j e^{c_j})$$

Z를 최소화하기 위해 위 식을 c_j에 관해 미분한 결과가 0이 되도록 c_j를 선택하면 $c_j = \frac{1}{2} \ln(\frac{w_+^j}{w_-^j})$를 얻는다. 이를 다시 Z의 식에 대입하면 $Z = 2\sum_j \sqrt{(w_+^j w_-^j)}$가 되고 이 값을 최소로 하는 단어를 h로 선택한다. 이 때 c_j의 부호는 문서에 단어가 포함되어 있는지의 여부에 따라 주제와의 관련성 여부를 가리킨다. 즉 어떤 단어가 포함되어 있는 문서 중 주제와 관련된 문서의 수가 많으면 c_1의 값이 양수가 되고, 주제와 관련없는 문서의 수가 많으면 음수가 된다. 또 $|c_j|$는 이렇게 판단할 때의 확신 정도를 나타내게 된다.

실제 구현에 있어 w_+^j, w_-^j의 값이 0이거나 0에 매우 가까운 값일 경우 Z와 c_j의 계산에 문제가 발생할 수 있다. 또 $|c_j|$가 지나치게 큰 값을 가질 경우 데이터 분포 D의 갱신시 지나치게 빠른 수렴 때문에 과적화 문제가 발생할 수 있다. 이러한 문제를 해결하기 위해 w_+^j와 w_-^j에 아주 작은 양의 상수인 E을 더해 Z와 c_j 값을 계산한다. 즉

$$Z = \sum_j w_+^j \sqrt{\left(\frac{w_-^j + E}{w_+^j + E}\right)} + w_-^j \sqrt{\left(\frac{w_+^j + E}{w_-^j + E}\right)}$$

$$c_j = \frac{1}{2} \ln\left(\frac{w_+^j + E}{w_-^j + E}\right)$$

이 때 $|c_j|$는 $\frac{1}{2} \ln(\frac{1}{E})$를 상한값으로 갖게 되어 위에서 언급한 문제를 해결할 수 있다. 이 알고리즘을 사용할 경우 4.1절의 방법과 거의 같은 계산량을 가지

면서도 빠른 수렴속도와 개선된 예측 성능을 나타낸다. 특히 학습 데이터의 수가 많은 경우 학습에 필요한 시간을 많이 줄일 수 있으며 분류 성능도 상당히 좋아지는 것을 볼 수 있다. 그러나 학습 데이터의 수가 적은 경우에는 과적화 문제가 발생하여 일반화 성능면에서 문제가 생긴다. 이러한 문제를 완화하기 위해 다음에서 설명하는 것과 같은 방법을 에이다부스트 알고리즘에 추가하였다.

4.3 반복적 에이다부스트 방법

앞에서 언급한 바와 같이 에이다부스트는 약 가설을 위원회에 추가함에 따라 학습 데이터에 대한 에러가 계속 감소하게 하기는 하지만 그 감소율은 점차 줄어든다. 그리고 일반화 에러는 학습 데이터에 대한 에러가 계속 줄어들더라도 어느 정도 이후에는 더 이상 줄어들지 않고 오히려 증가하는 과적화 현상을 볼 수 있다. 그렇다면 에이다부스트 알고리즘을 통해 어느 정도 에러를 줄인 후 데이터 분포를 임의로 초기화하는 방식으로 에이다부스트를 반복적으로 수행하면 보다 다양한 분포에 대한 학습을 하게 되어 개선된 일반화 에러를 나타내지 않을까? 다시 말해 다양한 분포를 갖는 데이터에 대해 여러번의 에이다부스트 방법을 수행한 후 생성된 모든 약 가설들간의 투표를 실시하여 최종적 예측을 수행하는 방법이 일반화 에러를 줄여주리라는 생각에서 다음과 같은 간단한 알고리즘을 에이다부스트에 추가적으로 적용하였다.

입력 : 에이다부스트 알고리즘
에이다부스트의 라운드수 T,
에이다부스트 수행 횟수 U

1. T 라운드만큼 에이다부스트를 수행하여 약 가설들을 생성한다.
2. 데이터 분포 D를 연속 푸아송 분포(continuous Poisson distribution)에 따라 임의로 초기화한후 합이 1이 되도록 정규화한다.
3. 1, 2의 과정을 U번 반복한다.
4. 학습된 모든 가설들간의 가중치가 부여된 투표를 실시하는 최종 가설을 출력한다.

표 4 반복적 에이다부스트

최초의 데이터 분포를 나타내는 $D_1(i)$는 에이다부스트와 마찬가지로 균일하게

$1/m$의 값이 부여된다. 정해진 라운드 수만큼 에이다부스트를 수행한 후 다시 D를 초기화할 때 연속 푸아송 분포를 사용하는데 이것은 새로운 데이터 집합을 표본 추출에 의해 생성하여 다양한 분포를 갖는 데이터 집합에 대해 학습을 수행하는 것을 가중치만을 변경하여 모방하기 위해서이다. 하나의 데이터가 임의 복원 표본 추출을 통해 새로 생성된 데이터 집합에 나타나는 횟수는 이산 푸아송 분포를 따른다고 알려져 있는데 [webb] 가중치를 연속 푸아송 분포로 초기화함으로써 이를 모방하는 것이다. 이러한 분포를 갖는 가중치는 다음과 같은 식을 사용하여 생성한다.

$$poisson() = - \log\left(\frac{random(1\cdots 999)}{1000}\right)$$

여기서 $random(\min\cdots\max)$는 $\min$과 $\max$ 사이의 임의의 값을 반환하는 함수이다. 이 알고리즘에서 에이다부스트의 수행 라운드 수와 몇번 데이터 분포를 초기화하여 에이다부스트를 수행할 것인가의 문제는 실험을 통해 결정하였다.

이 알고리즘은 2장에서 언급한 멀티부스트 알고리즘의 가중치 재초기화 방법을 단순화한 것으로 TREC과 같이 학습 데이터의 수가 작은 데이터 집합에 대해 에이다부스트의 과적화 문제를 완화하면서 일반화 에러를 줄이기 위해 도입한 것이다. 위와 같이 데이터 분포를 임의로 여러번 초기화하여 부스팅을 수행함으로써 데이터 분포에 따른 분산을 줄이게 되어 일반화 에러가 줄어드는 것을 알 수 있다.

제5장 실험 및 평가

5.1 실험 데이터

이번 실험에서 사용한 데이터는 Reuter 뉴스 기사 데이터와 TREC_8 adhoc 데이터이다. Reuter 데이터는 10개의 주제에 대한 관련성 여부가 표시된 문서들의 집합으로 학습 데이터로 8762개의 문서를, 테스트 데이터로 3009개의 문서를 사용하여 실험하였다. 또 8754개의 항 (단어 또는 어구)을 TF-IDF 방법을 사용하여 추출, 사용하였다. 이 데이터는 4.1절에 소개한 문서 여과를 위한 에이다부스트 방법과 4.2절의 실수값을 출력하는 가설을 이용한 에이다부스트 학습 알고리즘의 에러 감소 속도 및 최종 에러를 비교하기 위해 사용하였다. 이 문서 집합에서 각 주제와의 관련성을 갖는 문서 수와 관련이 없는 문서의 수를 다음 표에서 제시하고 있다.

주제	학습 데이터		테스트 데이터	
	관련있는 문서 수	관련없는 문서수	관련있는 문서 수	관련없는 문서수
acq	1483	7279	640	2369
corn	133	8629	36	2973
crude	334	8428	156	2835
earn	2706	6056	1034	1966
grain	370	8392	125	2884
interest	275	8487	97	2912
money-fx	428	8334	3009	131
ship	180	8582	80	2929
trade	303	8459	102	2907
wheat	185	8577	54	2955

표 5 Reuter 뉴스 데이터 집합

두번째 데이터는 TREC-8 ad hoc collection 데이터로, 1992년 Financial Times 기사 모음을 학습 데이터로 사용하고 1993년과 1994년 기사를 테스트 데이터로 사용한다. 전처리 과정에서는 300개의 stop 어휘를 제거하고 stemming을 수행한 후 TF-IDF를 사용하여 3000개의 항을 추출하였다. 이 데이터는 351 400의 50개 주제와 관련된 문서들의 집합으로 모든 문서에 관련성 여부가 표시되어 있는 것이 아니라 문서 집합의 일부에만 표시되어 있다. 따라서 이 실험에서는 관련성 여부

가 표시된 데이터만을 추출하여 사용하였다. 다음 표에 각 주제에 대해 관련성이 표시된 문서의 수, 관련이 있는 것으로 표시된 문서의 수, 관련이 없는 것으로 표시된 문서 수 등을 제시하였다.

주제	학습 데이터 수	관련있는 문서 수	관련없는 문서수	테스트 데이터 수	관련있는 문서수	관련없는 문서수
351	160	11	149	446	17	429
352	499	55	444	1109	182	927
353	152	15	137	267	24	243
354	212	21	191	319	41	278
355	83	1	82	208	1	207
356	255	8	247	484	7	477
357	143	24	119	229	40	189
358	53	0	53	106	0	106
359	186	4	182	500	17	483
360	66	2	64	152	12	140
361	181	0	181	412	1	411
362	61	0	61	129	3	126
363	164	2	162	506	3	503
364	113	1	112	210	2	208
365	142	8	134	205	3	202
366	143	4	139	465	15	450
367	149	12	137	263	25	238
368	87	87	0	262	5	257
369	104	1	103	232	0	232
370	107	15	92	232	6	226
371	125	1	124	407	1	406
372	99	3	96	247	12	235
373	144	2	142	359	10	349
374	97	19	78	264	53	211
375	106	9	97	198	4	194
376	363	13	350	591	15	576
377	136	5	131	270	10	260
378	467	51	416	813	36	777
379	118	0	118	292	0	292
380	104	0	104	273	1	272
381	136	5	131	328	2	326
382	70	2	68	170	4	166
383	114	15	99	358	65	293
384	57	1	56	131	3	128
385	125	9	116	213	25	188
386	107	1	106	235	3	232
387	55	4	51	220	9	211
388	96	3	93	170	12	158
389	237	41	196	458	98	360
390	151	2	149	332	49	283
391	221	55	166	673	106	567

주제	학습 데이터 수	관련있는 문서 수	관련없는 문서수	테스트 데이터 수	관련있는 문서수	관련없는 문서수
392	189	27	162	322	21	301
393	244	4	240	305	5	300
394	151	0	151	396	5	391
395	186	36	150	341	51	290
396	101	6	95	283	3	280
397	123	2	121	325	5	320
398	145	10	135	191	7	184
399	81	4	77	143	6	137
400	112	34	78	149	16	133

표 6 TREC_8 ad hoc collection **데이터**

위의 표에서 보는 바와 같이 이 데이터 집합에서는 각 주제에 대한 관련성 여부가 표시된 문서의 수가 적고 그 중 관련성이 있는 것으로 표시된 문서의 수는 매우 희박한 정도이다. 따라서 이와 같이 적은 데이터를 사용하여 학습을 수행하는데는 과적화의 문제 등 많은 문제가 수반된다.

5.2 실험 결과 및 평가

이 논문에서는 두가지 실험을 수행하였다. 먼저 기존의 에이다부스트 알고리즘과 실수값을 출력하는 가설을 이용한 에이다부스트 방법의 에러, 회수율, 정확률을 비교하는 실험을 Reuter 데이터에 대해 수행하였다. 그림 1에서는 주제 acq에 대한 AdaBoost-tf (4.1절의 알고리즘)과 real-AdaBoost (4.2절의 알고리즘)의 학습 에러와 테스트 에러를 보여주고 있다.

이 실험에서는 AdaBoost-tf의 경우 10,000개의 약 가설을 학습하였고, real-AdaBoost의 경우는 1,000개의 약 가설만을 학습시켰다. 그러나 최종 에러에 있어 real-AdaBoost가 학습 에러와 테스트 에러 모두 작은 값을 보임을 알 수 있다. 또 테스트 에러가 거의 변화하지 않는 것으로 보이는 지점에 도달하기 위해 AdaBoost-tf는 4,000여회 이상의 학습이 필요하지만 real-AdaBoost의 경우 430 여회의 학습만으로 보다 개선된 테스트 에러를 나타냄을 알 수 있다. 즉 real-AdaBoost가 약 10배 정도의 학습 시간을 단축하면서도 테스트 에러 역시 보다 작게 나타나는 현상을 보이고 있는 것이다. 여기서 한가지 언급할 것은 에이다부스트 학습 알고리즘의 경우 학습 데이터에 대한 에러는 0에 이르기까지 계속해서 감소하는 경향을 보이지만 테스트 데이터에 대한 에러는 어느 시점 이후에는 거의 변화가 없는 것을 알 수 있다. 즉 학습 데이터에 대한 에러의 감소가 일반화 에러를 더 이상 향상시키지 못하는 지점 이후의 학습은 분류 시스템의 성능 개선에 별 도움이 되지 않으므로 학습 시간이 오래 걸린다는 문제를 갖는 에이다부스트의 단점을 생각할 때 적절한 지점에서 학습을 중단하는 것이 필요하다.

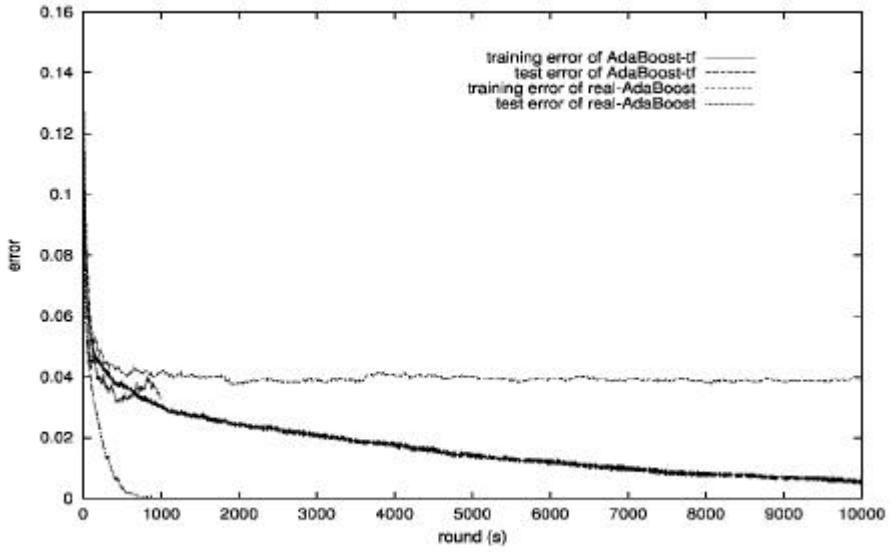

그림 1 AdaBoost-tf와 real-AdaBoost의 학습 데이터와 테스트 데이터에 대한 에러. 여기서 AdaBoost-tf는 4.1절에 제시된 기존의 문서 여과를 위한 에이다부스트 알고리즘이고, real-AdaBoost는 4.2절의 실수값을 출력하는 가설을 이용한 에이다부스트 알고리즘이다.

그림 2와 3에서는 AdaBoost-tf와 real-AdaBoost의 학습 데이터와 테스트 데이터에 대한 회수율과 정확률을 각각 보여주고, 이 두 알고리즘의 비교를 위해 그림 4에서는 테스트 데이터에 대한 회수율과 정확률을 함께 보여주고 있다. 역시 AdaBoost-tf는 10,000회를, real-AdaBoost는 1,000회를 수행한 것인데 특히 그림 3에 나타난 real-AdaBoost의 경우, 학습 데이터에 대한 정확률이 1,000회의 학습을 수행하기 전에 1에 이르고, 테스트 데이터에 대해서는 회수율이 88%, 정확률이 96%이상에 달하는 것을 알 수 있다. 그림 4에 나타난 것과 같이 역시 real-AdaBoost가 AdaBoost-tf보다 학습 데이터와 테스트 데이터 모두에 대해 회수율과 정확률 모두 좋은 값을 나타내고 있다.

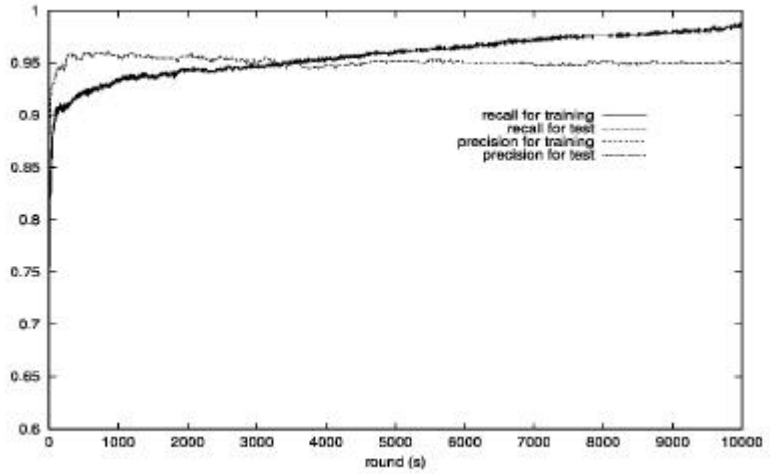

그림 2 AdaBoost-tf의 학습데이터와 테스트데이터에 대한 회수율과 정확률

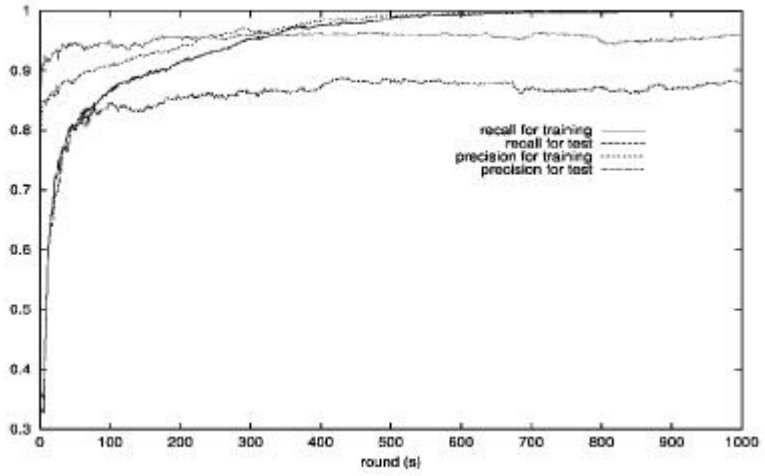

그림 3 real-AdaBoost의 학습 데이터와 테스트 데이터에 대한 회수율과 정확률

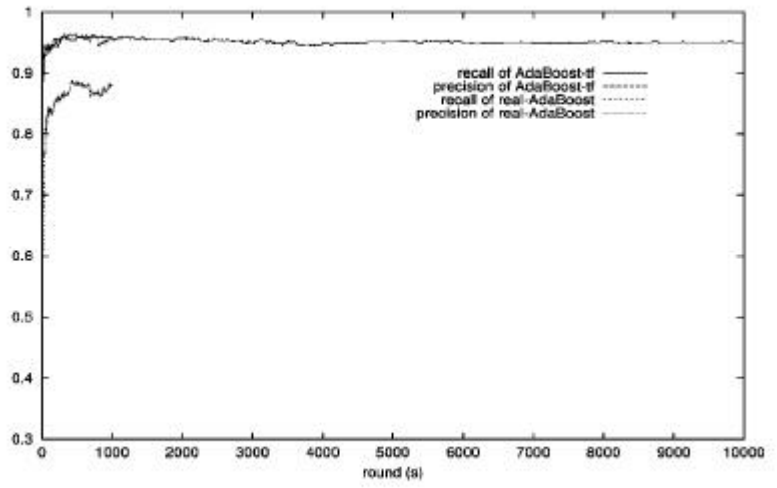

그림 4 AdaBoost-tf와 real-AdaBoost의 테스트 데이터에 대한 회수율과 정확률 비교

그림 1 4에서 보는 바와 같이 real-AdaBoost가 AdaBoost-tf보다 좋은 성능을 나타내는 이유로는 다음과 같은 두가지를 생각해 볼 수 있다. 먼저 real-AdaBoost에서는 이진 값이 아닌 실수값을 출력하는 가설과 이의 학습에 적절한 에러 상한을 사용하기 때문에 AdaBoost-tf보다 정확한 에러 조절이 가능하다. 두번째 이유로는 AdaBoost-tf에서는 어떤 단어가 문서에 포함되어 있으면 1, 그렇지 않으면 -1로 가설의 출력값이 고정되고 이 값에 대한 확신 정도는 가설의 가중치인 α로 표시하는데 반해, real-AdaBoost에서는 단어가 포함되어 있을 경우와 포함되어 있지 않을 경우에 대한 확신 정도를 따로 계산하여 분류에 사용하므로 최종 결정 과정에서 보다 정확한 결정을 내릴 수 있게 된다. 다시 말해 AdaBoost-tf에서는 어떤 단어가 문서에 포함되어 있는 경우와 포함되어 있지 않은 경우 모두 $|\alpha * h_s|$ 가 같은 값을 갖지만 real-AdaBoost에서는 서로 다른 값을 가질 수 있기 때문에 보다 정확한 예측이 가능하게 되는 것이다.

다음으로 두번째 실험에서는 TREC-8 문서 여과 과제의 경기 규칙에 입각하여

real-AdaBoost와 i-AdaBoost (4.3절의 알고리즘)을 사용한 실험을 수행하였다. 이 실험에서 real-AdaBoost는 100개의 가설을 학습해 가면서 각 가설을 위원회에 추가한후 테스트 데이터에 대한 LF1 값을 측정하고 LF1 값이 최대가 되는 시점을 찾아 표시하였다. 이러한 과정을 50개 주제에 대해 시행하고 가장 좋은 LF1 값을 모두 합하여 최종 결과로 삼았다. 한편 i-AdaBoost에서는 주어진 라운드 수만큼 에이다부스트를 수행하고 임의로 데이터에 대한 가중치를 초기화하여 다시 에이다부스트를 수행하는 것을 정해진 수만큼 반복하면서 테스트 데이터에 대한 LF1 값을 측정하였다. 이때 에이다부스트의 수행 횟수는 30, 50, 70, 100 회를 실험하였고 반복 횟수는 5회로 하였다. 이때 최대 LF1 값을 나타낸 50번의 에이다부스트를 수행한 실험에서 각 주제에 대한 최대 LF1 값과 그 때의 라운드 수를 real-AdaBoost의 경우와 함께 표 7에 나타내었다. 또 표 8에서는 에이다부스트 수행 횟수 30, 50, 70, 100일 때의 결과를 보이고 있다.

주 제	real-AdaBoost		i-AdaBoost	
	라운드 수	LF1	라운드 수	LF1
351	25	63	62	12
352	21	20	25	63
353	9	13	21	20
354	16	0	68	14
355	9	1	16	0
356	3	34	9	1
357	1	0	3	34
358	1	0	1	0
359	2	1	1	0
360	1	0	2	1
361	1	0	1	0
362	1	0	1	0
363	1	0	1	0
364	1	0	1	0
365	1	0	1	0
366	7	6	107	3

주제	real- AdaBoost		i- AdaBoost	
	라운드 수	LF1	라운드 수	LF1
367	1	0	7	6
368	17	- 2	1	0
370	1	0	1	0
371	17	0	17	0
372	2	6	2	6
373	1	0	1	0
374	1	0	202	33
375	1	0	1	0
376	1	0	1	0
377	1	0	1	0
378	1	0	1	0
379	1	0	1	0
380	1	0	1	0
381	1	0	1	0
382	3	0	3	0
383	1	84	1	84
384	1	0	1	0
385	1	0	104	22
386	18	0	18	0
387	1	0	1	0
388	1	0	1	0
389	69	50	63	50
390	1	0	1	0
391	3	83	3	83
392	2	9	2	9
393	1	0	1	0
394	1	0	1	0
395	16	3	270	16
396	1	0	1	0
397	2	0	2	0
398	1	0	154	7
399	1	0	1	0
400	9	- 10	9	- 10
total		374		454

표 7 real- AdaBoost**와** i- AdaBoost**의** LF1 **측정치**

주제	T =30	T =50	T =70	T =100
351	18	12	13	13
352	63	63	63	63
353	20	20	20	20
354	13	14	13	13
355	0	0	0	0
356	1	1	1	1
357	34	34	34	34
358	0	0	0	0
359	0	0	0	0
360	1	1	1	1
361	0	0	0	0
362	0	0	0	0
363	0	0	0	0
364	0	0	0	0
365	1	0	0	0
366	3	3	3	3
367	6	6	6	6
368	0	0	0	0
369	- 2	0	- 2	- 2
370	0	0	0	0
371	0	0	0	0
372	6	6	6	6
373	0	0	0	0
374	33	33	24	33
375	0	0	0	0
376	0	0	0	0
377	0	0	0	0
378	0	0	0	0
379	0	0	0	0
380	0	0	0	0
381	0	0	0	0
382	0	0	0	0
383	87	84	84	84
384	0	0	0	0
385	12	22	5	11
386	0	0	0	0
387	0	0	0	0

주제	T=30	T=50	T=70	T=100
388	0	0	0	0
389	44	50	53	52
390	10	0	0	0
391	83	83	83	83
392	9	9	9	9
393	0	0	0	0
394	0	0	0	0
395	13	16	17	10
396	0	0	0	0
397	0	0	0	0
398	7	7	9	9
399	0	0	3	0
400	- 10	- 10	- 10	- 10
total	442	454	435	439

표 8 부스팅 라운드수에 따른 결과 비교

표 7에서 알 수 있는 바와 같이 i-AdaBoost는 real-AdaBoost에 비해 LF1의 총 점수에서 큰 차이를 보이고 있다. 두 알고리즘은 대부분의 주제에 대해 같은 LF1 값을 나타내지만 374나 385와 같은 주제에 대해 i-AdaBoost가 훨씬 좋은 점수를 얻고 있는데 이것은real-AdaBoost의 수렴성을 그대로 지니면서 데이터에 대한 가중치를 임의로 초기화함으로써 다양한 데이터 분포에 대한 학습을 수행한 결과이다. 또 여기서 첫번째 라운드에서 최고의 값을 얻고 있는 주제들이 나타나 있는데 이것은 학습이 진행되지 않은 것이 아니라 그 주제에 관련성이 있는 것으로 표시된 문서의 수가 0이거나 아주 작은 경우로 이 때는 더 이상 학습을 진행하는 것이 무의미하므로 이러한 결과가 나온 것이다. 표 8에 제시되어 있는 i-AdaBoost에 대한 실험 결과에서는 i-AdaBoost에서 한번의 에이다부스트를 수행하는 라운드 수에 따라 최종 LF1 값이 달아짐을 알 수 있는데 여기서 실험한 30, 50, 70, 100 회 중 50회 수행한 것이 가장 좋은 결과를 보였고 다른 경우 모두 real-AdaBoost의 경우보다는 높은 점수를 나타내고 있다.

다음으로 표 9에서는 TREC-8 여과 과제에 참가한 7개 팀의 각 주제에 대한 LF1 값과 최대LF1 값, 그리고 비교를 위해 i-AdaBoost (T=50)의 LF1 값을 보여주고 있다. 여기서 LF1의 총점에 있어 i-AdaBoost (T=50)가 가장 높은 점수를

나타내고 있음을 알 수 있다.

주제	i- AdaBoost	PLt8f1	PLt8f2	CL99bfl1	pirc9BF1	Mer8BaLF1	AntBatch1	Scai8Ft	Maximal
351	12	22	25	20	25	- 320	15	0	51
352	63	82	115	12	130	100	9	149	570
353	20	32	42	4	28	21	- 1001	0	87
354	14	- 9	- 9	- 2	- 7	- 15	- 261	0	147
355	0	0	0	0	0	0	- 324	0	3
356	1	7	7	- 5	4	- 345	0	- 2	45
357	34	47	41	34	59	46	- 43	3	183
358	0	0	0	0	0	0	0	0	6
359	0	- 5	- 5	0	0	0	- 992	- 8	90
360	1	0	0	0	0	3	- 1002	0	36
361	0	0	0	- 6	0	0	0	0	6
362	0	0	0	1	0	0	0	- 2	15
363	0	0	0	- 2	- 4	- 4	0	0	12
364	0	0	0	0	0	- 2	- 496	0	6
365	0	- 16	- 28	- 2	4	0	3	0	9
366	3	0	0	4	3	22	- 1002	0	51
367	6	0	5	0	4	4	- 28	0	90
368	0	0	0	0	0	0	- 55	0	15
369	0	0	0	0	0	0	0	0	0
370	0	- 20	- 20	- 2	- 13	- 4	- 389	0	93
371	0	0	0	0	0	0	0	0	3
372	6	3	3	- 6	1	0	0	- 4	36
373	0	0	0	13	0	0	0	0	36
374	33	5	5	- 4	- 19	12	- 152	- 2	177
375	0	- 2	1	1	3	3	3	0	30
376	0	3	3	- 9	0	0	- 366	0	51
377	0	6	6	9	2	3	0	0	30
378	0	- 25	- 14	- 12	- 12	- 11	- 11	5	132
379	0	0	0	- 8	0	0	0	0	0
380	0	0	0	0	0	0	0	0	3

주제	i-AdaBoost	PLt8f1	PLt8f2	CL99bfl1	pirc9BF1	Mer8BaLF1	AntBatch1	Scai8Ft	Maximal
381	0	0	0	-6	0	0	0	0	6
382	0	-2	-2	7	-2	0	-1002	0	12
383	84	97	113	-2	72	27	-7	-2	201
384	0	0	0	0	0	0	0	0	9
385	22	9	14	24	28	19	11	0	81
386	0	0	0	0	-2	0	0	0	18
387	0	-10	-10	-6	-1	3	-1002	0	27
388	0	0	0	-3	0	0	-323	0	42
389	50	145	113	122	76	31	176	218	387
390	0	-4	-4	0	0	3	-2	-4	204
391	83	36	-18	57	-35	-16	-22	-138	381
392	9	-32	-32	-5	2	-5	-21	4	96
393	0	7	7	-2	1	3	-14	0	15
394	0	0	0	-8	0	0	0	0	15
395	16	-8	-1	-6	-50	-1	-115	-9	186
396	0	1	1	-6	4	0	0	0	15
397	0	-2	-2	0	0	-6	0	0	15
398	7	9	9	-1	0	4	-3	0	30
399	0	0	0	4	3	0	0	0	27
400	-10	-13	-8	3	-9	5	-20	0	48
total	454	363	357	212	295	-420	-8436	208	3828

표 9 TREC-8 참가자들과 i-AdaBoost의 LF1 비교

i-AdaBoost가 다른 학습 방법들에 비해 높은 LF1 값을 갖는 것은 우선 대부분의 주제에 대해 음수값을 나타내지 않는다는데 원인이 있는데 이것은 i-AdaBoost가 사용하는 약 가설이 가지는 높은 바이어스 때문에 어느 정도의 성능을 보장하기 때문이다. 또 한가지 주목할 만한 점은 i-AdaBoost가 데이터에 대한 가중치의 초기화를 통해 real-AdaBoost보다 좋은 값을 얻게 하는 주제에 대해서 다른 방법들에 비해 높은 점수를 나타내는 경향을 보인다. 이것은 다양한 데이터 분포에 대한 학습이 일반화 에러를 줄여 줄수 있음을 입증하는 또 하나의 증거가 된다. 한편 352번 주제와 같이 비교적 많은 관련성 있는 문서를 포함한 주제에 대해서는 최고의 성능을 나타내지 못하고 있는데 그 이유는 i-AdaBoost가 많은 문서에 대해 학습을 잘 수행하지 못하기 때문이 아니라 이 정도 수의 적은 문서에 대해서는 학습 데이터에 대한 과적화 현상이 발생하여 낮은 일반화 에러를 얻기 힘들기 때

문이다. 마지막으로 이번 실험을 수행한 환경을 표 10에 제시하였다.

	AdaBoost-tf	real-AdaBoost	i-AdaBoost
시스템	**펜티움** III 500mhz, Windows98		
라운드수	100	100	30*5, 50*5, 70*5, 100*5
실행시간	5**분**	5**분**	7**분**, 9**분**, 11**분**, 15**분**

표 10 **실험 환경**

제6장 결론 및 토의 사항

이 논문에서는 비교적 새로운 기계 학습 방법인 에이다부스트 학습 방법을 사용하여 문서 검색 대회인 TREC의 문서 여과 문제에 적합한 문서 여과 시스템을 생성하고자 하였다. 먼저, Freund와 Schapire의 에이다부스트 학습 방법을 문서 여과 문제에 적용한 알고리즘을 구현하고 이의 학습 시간이 오래 걸리는 문제를 개선하기 위해 이진 값을 출력하는 가설이 아닌 실수값을 출력하는 가설을 이용하는 에이다부스트 알고리즘을 도입하였다. 또 TREC 데이터가 가지는 특성에 대해 보다 뛰어난 여과 성능을 나타내도록 하기 위해 연속 푸아송 분포에 따라 임의로 초기화된 여러 개의 데이터 분포에 대해 에이다부스트를 수행하는 알고리즘을 적용하였다. 실험은 먼저 Reuter 뉴스 기사 데이터를 사용하여 이진 가설을 사용하는 기존의 에이다부스트에 비해 실수값을 출력하는 가설을 이용한 에이다부스트 알고리즘이 10배에 가까운 속도의 향상과 개선된 에러를 나타냄을 보였다. 다음으로 이와 같이 개선된 에이다부스트 방법을 TREC에 적용하여 실험하고 이의 결과를 향상하기 위해 데이터 분포를 임의로 변경하여 반복해서 수행하는 에이다부스트 알고리즘을 사용하였다. 문서 여과 시스템의 성능에 대한 평가 척도로는 TREC에서 공식적으로 사용하고 있는 LF1 척도를 사용하였고 그 결과 이 논문에서 사용한 부스팅 알고리즘이 가장 높은 값을 나타내었다.

이번 연구와 관련하여 앞으로 더욱 많은 연구가 필요한 몇가지 사항들을 살펴보면 다음과 같다.

1. 에이다부스트를 이용하여 학습을 진행함에 따라 발생하는 과적화 문제와 관련하여 어떻게 일반화 에러를 최소화하는 시점에서 학습을 중단할 것인가의 문제를 해결할 필요가 있다. 즉 학습 데이터에 대한 에러가 감소한다고 해서 일반화 에러가 함께 감소하는 것이 아니기 때문에, 또 일반화 에러의 감소가 규칙적으로 진행되는 것이 아니기 때문에 적절한 시점에서 학습을 중단하는 것이 필요하다. 이 논문에서는 실험을 통해 테스트 데이터에 대해 가장 작은 에러를 보이는 라운드를

찾아 그때까지의 가설만을 최종 투표에 사용하였으나 범용 문서 여과 시스템의 개발을 위해서는 보다 체계적인 방법이 필요하리라 생각한다.

2. TREC 데이터의 항 수가 문서 여과 시스템의 성능에 미치는 영향에 대해 자세히 알아볼 필요가 있다. 이 논문에서는 TF-IDF 방법을 사용하여 3,000개의 항을 갖는 학습 데이터와 테스트 데이터를 생성하였으나 데이터에 포함된 항 수를 달리 하여 실험함으로써 항 수와 최종 결과와의 관계를 살펴볼 필요가 있다.

3. i-AdaBoost 알고리즘의 적용에 있어 한번 에이다부스트를 수행하는 라운드 수와, 몇번의 에이다부스트를 수행할 것인가의 문제를 보다 체계적으로 결정하는 방법이 필요하다. 위의 실험에서 볼 수 있듯이 한번의 에이다부스트 수행시 몇개의 라운드를 학습하느냐에 따라 다른 결과가 나타나고 이를 주어진 문제에 최적화된 값으로 결정하는 체계적 방법이 필요한 것이다.

지금까지 살펴본 에이다부스트를 사용한 문서 여과 시스템의 경우 최종 결정에 사용되는 각 가설이 어떤 단어가 문서에 포함되어 있는가의 여부이므로 이 시스템의 사용자가 시스템의 결정규칙을 쉽게 이해할 수 있고 각 가설이 판단 기준이 되는 단어와 이에 관련된 확신정도, 이 두개의 값만을 저장하기 때문에 의사결정트리나 인공신경망에 비해 비교적 적은 메모리를 사용하여 실제 구현에 많은 문제를 초래하지 않는다. 이와 같이 구현과 이해가 용이하고 우수한 성능을 나타내는 학습 방법에 대한 보다 많은 연구를 통해 일반 사용자들이 보다 쉽게 문서를 검색할 수 있는 방법들이 개발되기를 바란다.

참 고 문 헌

[abney99] Steven Abney, Robert E. Schapire, and Yoram Singer. Boosting applied to Tagging and PP attachment. *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

[bauer99] Bauer E., and Kohavi R.. An empirical comparison of voting classification algorithms: bagging, boosting, and variants. *Machine Learning*, volume 36 Issue 1, pages 105 139, 1999.

[breiman96a] L. Breiman. Bagging predictors. *Machine Learning*, volume 24, 123 140, 1996.

[breiman96b] L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460. Berkeley, CA: University of California: Department of Statistics, 1996.

[breiman97] L. Breiman. Arcing the edge. Technical Report 486. Berkeley, CA: University of California: Department of Statistics. 1996.

[freund96a] Y. Freund, R. E. Schapire. Experiments with a new boosting algorithm. *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148 156, 1996.

[freund96b] Y. Freund, R. E. Schapire. Game theory, on-line prediction and boosting, *Proc. 9th Annu. Conf. on Comput. Learning Theory*, pp. 325-332, ACM Press, New York, NY, 1996.

[freund97] Yoav Freund, Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, volume 55, pages 119 139, 1997.

[freund99] Yoav Freund, Robert. E. Schapire. The large margin classification using the perceptron algorithm. *Machine Learning*, volume 37, Issue 3, pages 277 296, 1999.

[friedman] J. Friedman, T. Hastie, and R. Tiibshirani. Additive logistic regression: A statistical view of boosting. Annals of Statistics, to appear.

[hull96] David Hull, Jan Pedersen, and Hinrich Schutze. Method combination for document filtering. *Proceeding of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 279 288, 1996.

[lewis92] David Lewis. An evaluation of phrasal and clustered representations on a text categorization task. *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37 50, 1992.

[lewis94] David Lewis and William Gale. A sequential algorithm for training text classifiers. *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3 12, 1994.

[lewis96] David Lewis, Robert Schapire, James Callan, and Ron Papka. Training algorithm for linear text classifiers. *Proceeding of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298 306, 1996.

[neuralnet] Simon Haykin, Neural Networks. Prentice-Hall, Inc, 1999.

[quinlan96] J. R. Quinlan. Bagging, Boosting, and C4.5. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725 730, 1996.

[schapire98a] R. E. Schapire, Y. Freund, P. Bartlett, W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, volume 26, 1651 1686, 1998.

[schapire98b] Robert E. Schapire, Yoram Singer, and Amit Singhal. Boosting and Rocchio applied to text filtering. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Query and Profile Modification, pp. 215-223, 1998.

[schapire99a] Robert E. Schapire, Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, volume 37, Issue 3, 1999.

[schapire99b] Robert E. Schapire, and Yoram Singer. BoosTexter: a boosting-based system for text categorization. *Machine Learning*, 1999.

[webb] Geoffrey I. Webb. MultiBoosting: A technique for Combining Boosting and Wagging. *Machine Learning*, to appear.

[yang94] Yiming Yang. Expert network: effective and efficient learning from human decisions in text categorization and retrieval. *Proceeding of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 13 22, 1994.

[yang95] Yiming Yang. Noise reduction in a statistical approach to text categorization. *Proceeding of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 25 6 263, 1995.

Abstract

Text filtering is a task of deciding whether a document has relevance to a specified topic. As Internet and Web becomes wide-spread and the number of documents delivered by e-mail explosively grows the importance of text filtering increases as well. The aim of this thesis is to improve the accuracy of text filtering systems by using machine learning techniques. We apply AdaBoost algorithms to the filtering task. An AdaBoost algorithm generates and combines a series of simple hypotheses. Each of the hypotheses decides the relevance of a document to a topic on the basis of whether or not the document includes a certain word. We begin with an existing AdaBoost algorithm which uses weak hypotheses with their output of 1 or -1. Then we extend the algorithm to use weak hypotheses with real-valued outputs to improve error reduction rate and final filtering performance. Next, we attempt to achieve further improvement in the AdaBoost's performance by first setting weights randomly according to the continuous Poisson distribution, executing AdaBoost, repeating these steps several times, and then combining all the hypotheses learned. This has the effect of mitigating the overfitting problem which may occur when learning from a small number of data. Experiments have been performed on the real document collections used in TREC-8, a well-established text retrieval contest. This dataset includes Financial Times articles from 1992 to 1994. The experimental results show that AdaBoost with real-valued hypotheses outperforms that with binary-valued hypotheses, and that AdaBoost iterated with random weights further improves filtering accuracy. Comparison results of all the participants of the TREC-8 filtering task are also provided.

감사의 글

지난 2년을 돌아보면 시간의 빠름을 새삼 깨닫는다. 어색한 기분으로 연구실에 인사하러 왔던 일, 낯선 환경에 적응하기 위해 마음을 조리며 노력하던 일들이 어제의 일같이 생생한데 벌써 졸업을 눈 앞에 두고 있다. 아직 부족한 것이 많지만 그동안 많은 것을 배웠고 여러 경험을 했으며 약간의 자신감도 얻었다. 우선 이러한 것들을 경험할 수 있는 기회를 주신 김영택 교수님께 감사드리고, 그동안 여러 가지로 지도를 아끼지 않으신 장병탁 교수님께도 감사의 말씀을 드린다. 그리고 논문심사 위원장으로 수고해 주신 김형주 교수님과 그 밖의 모든 컴퓨터 공학과 교수님들에게 감사를 드린다. 이 논문의 작성 과정에서 실험 데이터 처리를 도와준 김유환, 논문의 편집과 교정을 자기 일처럼 수고해 준 정제균, 이지형, 그리고 내 동생 승희에게 고마움을 전한다. 또 적어도 1주일에 한번씩은 반찬과 빨래, 때때로 맛있는 음식을 가지고 못난 아들을 찾아주신 어머니, 아버지께도 감사의 말씀을 드리고 싶다. 지난 2년간 내가 책 읽는 것을 도와준 많은 자원봉사자 들에게도 감사하고 같은 연구실에서 함께 지내면서 정이 든 모든 선후배, 동기들에게도 감사의 말을 전한다. 그리고 무엇보다도 오늘의 나를 지금의 이 모습으로 있게 하신 하나님께 감사하지 않을 수 없다. 마지막으로 나를 아는 모든 사람, 아니 이 세상의 모든 사람에게 하나님의 은혜와 축복이 함께 하기를 기원한다.

2000년 1월 17일

한 상 윤